2.9 Complete Partial Orders and the Fixed-Point Theorem

2.9.1 Posets, Upper Bounds and Least Upper Bounds

Definition 10. *A **partial order**, or a **poset** (from partial order set) $(D, \sqsubseteq)$ consists of a set D and a binary relation $\sqsubseteq$ on D, written as an infix operation, which is*

- ***reflexive**, that is, $x \sqsubseteq x$ for any $x \in D$;*
- ***transitive**, that is, for any $x, y, z \in D$, $x \sqsubseteq y$ and $y \sqsubseteq z$ imply $x \sqsubseteq z$; and*
- ***anti-symmetric**, that is, if $x \sqsubseteq y$ and $y \sqsubseteq x$ for some $x, y \in D$ then $x = y$.*

*A partial order $(D, \sqsubseteq)$ is called a **total order** when $x \sqsubseteq y$ or $y \sqsubseteq x$ for any $x, y \in D$.*

Here are some examples of partial or total orders:

- $(\mathcal{P}(S), \subseteq)$ is a partial order, where $\mathcal{P}(S)$ is the set of subsets of a set S.
- $(\mathbb{N}, \leq)$, the set of natural numbers ordered by "less than or equal to", is a total order.
- $(\mathbb{N}, \geq)$, the set of natural numbers ordered by "larger than or equal to", is a total order.
- $(\mathbb{N} \cup \{\infty\}, \leq)$, the set of natural numbers plus infinity, where infinity is larger than any number, is a total order.
- $(\mathbb{Z}, \leq)$, the set of integer numbers, is a total order.
- $(\mathbb{R}, \leq)$, the set of real numbers, is a total order.
- $(S, =)$, a flat set S where the only partial ordering is the identity, is a partial order.
- $(S \cup \{\bot\}, \leq_\bot^S)$, a set S extended with a bottom element $\bot$ with the property that for any $a, b \in S \cup \{\bot\}$, $a \leq_\bot^S b$ if and only if $a = b$ or $a = \bot$, is a partial order. Such extensions of sets with bottom elements are common in denotational semantics (Section 3.7), where the intuition for the $\bot$ is "undefined". They are commonly written more simply as $S_\bot$.
- A partial order which is particularly important for denotational semantics in Section 3.7 is $(A \rightharpoonup B, \preceq)$, the set of partial functions from A to B partially ordered by the *informativeness relation* $\preceq$: given partial functions $f, g : A \rightharpoonup B$, f is "less informative than or as informative as" g, written $f \preceq g$, if and only if for any $a \in A$, it is either the case that $f(a)$ is not defined or both $f(a)$ and $g(a)$ are defined and $f(a) = g(a)$.
- If $(S_1, \leq_1)$ and $(S_2, \leq_2)$ are partial or total orders and $S_1 \cap S_2 = \varnothing$ then $(S_1 \cup S_2, \leq_1 \cup \leq_2)$ is a partial order, where $a(\leq_1 \cup \leq_2)b$ if and only if there is some $i \in \{1, 2\}$ such that $a, b \in S_i$ and $a \leq_i b$. This union partial order is typically not a total order; it is a total order only when $(S_1, \leq_1)$ is total and $S_2 = \varnothing$, or the other way around. More generally, if $\{(S_i, \leq_i)\}_{i \in I}$ is a family of partial orders such that $S_i \cap S_j = \varnothing$ for any $i \neq j \in I$, then $(\cup_{i \in I} S_i, \cup_{i \in I} \leq)$ is a partial order, where $a(\cup_{i \in I} \leq_i)b$ if and only if there is some $i \in I$ such that $a, b \in S_i$ and $a \leq_i b$.

Definition 11. *Given partial order $(D, \sqsubseteq)$ and a set of elements $X \subseteq D$, an element $p \in D$ is called an **upper bound (ub)** of X if and only if $x \sqsubseteq p$ for any $x \in X$. Furthermore, $p \in D$ is called the **least upper bound (lub)** of X, written $\sqcup X$, if and only if p is an upper bound and for any other upper bound q of X it is the case that $p \sqsubseteq q$.*

Note that upper bounds and least upper bounds may not always exist. For example, if $D = X = \{x, y\}$ and $\sqsubseteq$ is the identity relation, then X has no upper bounds. Least upper bounds may not exist even though upper bounds exist. For example, if $D = \{a, b, c, d, e\}$ and $\sqsubseteq$ is defined by $a \sqsubseteq c$, $a \sqsubseteq d$, $b \sqsubseteq c$, $b \sqsubseteq d$, $c \sqsubseteq e$, $d \sqsubseteq e$, then any subset X of D admits upper bounds, but the set $X = \{a, b\}$ does not have a least upper bound. Due to the anti-symmetry property, least upper bounds are unique when they exist, which justifies the notation $\sqcup X$ in Definition 11.

To make an analogy with mathemtical analysis, one can think of the lub of a set of elements as the "limit" of those elements. For example, if one works with $(\mathbb{R}, \leq)$ then the lub of the set $\{n/(n+1) \mid n \in \mathbb{N}\}$ coincides with its limit in the mathematical analysis sense, namely 1. Note that $([0, 1), \leq)$ does not admit lubs for all its subsets, e.g., $\{n/(n+1) \mid n \in \mathbb{N}\}$ does not have a lub.

2.9.2 Complete Partial Orders

Definition 12. *Given a poset $(D, \sqsubseteq)$, a **chain** in D is an infinite sequence $d_0 \sqsubseteq d_1 \sqsubseteq d_2 \sqsubseteq \cdots \sqsubseteq d_n \sqsubseteq \ldots$ of elements in D, also written using set notation as $\{d_n \mid n \in \mathbb{N}\}$. Such a chain is called **stationary** when there is some $n \in \mathbb{N}$ such that $d_m = d_{m+1}$ for all $m \geq n$.*

If D is finite then any chain is stationary. More generally, if for a given $x \in D$ there are only finitely many elements $y \in D$ such that $x \sqsubseteq y$, then any chain containing x is stationary.

Definition 13. *A poset $(D, \sqsubseteq)$ is called a **complete partial order (CPO)** if and only if any of its chains has a lub. Poset $(D, \sqsubseteq)$ is said to **have bottom** if and only if it has a minimal element. Such element is typically denoted by $\bot$, and the poset with bottom $\bot$ is written $(D, \sqsubseteq, \bot)$. CPOs with bottom are also called **bottomed complete partial orders (BCPO)**. If $\{d_n \mid n \in \mathbb{N}\}$ is a chain in $(D, \sqsubseteq)$, then we also let $\bigsqcup_{n \in \mathbb{N}} d_n$ or more simply $\sqcup d_n$ denote its lub $\sqcup\{d_n \mid n \in \mathbb{N}\}$.*

Let us follow up on the examples underneath Definition 10 and discuss which of them are cpos with botom and which are not:

- $(\mathcal{P}(S), \subseteq, \varnothing)$ is a BCPO.
- $(\mathbb{N}, \leq)$ has bottom 0 but is not complete: the chain $0 \leq 1 \leq 2 \leq \cdots \leq n \leq \cdots$ has no upper bound in $\mathbb{N}$.
- $(\mathbb{N}, \geq)$ is a CPO but has no bottom.
- $(\mathbb{N} \cup \{\infty\}, \leq, 0)$ is a BCPO with bottom 0: it is a CPO because any chain is either stationary, in which case its lub is obvious, or is unbounded by any natural number, in which case ∞ is its lub.
- $(\mathbb{Z}, \leq)$ is not a CPO and has no bottom.
- $(S, =)$ is always a CPO, and it is a BCPO if and only if S has only one element.

- $(S \cup \{\bot\}, \leq^S_\bot, \bot)$ is a BCPO. It is common to use the same notation $S_\bot$, used for the partial order $(S \cup \{\bot\}, \leq^S_\bot)$, to also denote its corresponding BCPO $(S \cup \{\bot\}, \leq^S_\bot, \bot)$.
- $(A \rightharpoonup B, \preceq, \bot)$, the set of partial functions $A \rightharpoonup B$ ordered by the informativeness relation $\preceq$, is a BCPO with bottom $\bot : A \rightharpoonup B$, the partial function which is undefined in each element of A.
- If $\{(S_i, \leq_i, \bot)\}_{i \in I}$ is a family of BCPOs with same bottom $\bot$ such that $S_i \cap S_j = \{\bot\}$ for any $i \neq j \in I$, then $(\cup_{i \in I} S_i, \cup_{i \in I} \leq, \bot)$ is also a BCPO.

2.9.3 Monotone and Continuous Functions

Definition 14. *If $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ are two posets and $\mathcal{F} : D \to D'$ is a function, then $\mathcal{F}$ is called **monotone** if and only if $\mathcal{F}(x) \sqsubseteq' \mathcal{F}(y)$ for any $x, y \in D$ with $x \sqsubseteq y$. If $\mathcal{F}$ is monotone, then we simply write $\mathcal{F} : (D, \sqsubseteq) \to (D', \sqsubseteq')$. Let $Mon((D, \sqsubseteq), (D', \sqsubseteq'))$ denote the set of monotone functions from $(D, \sqsubseteq)$ to $(D', \sqsubseteq')$.*

Exercise 22. *Recall that given a set S, $S_\bot$ denotes the poset $(S \cup \{\bot\}, \leq^S_\bot)$ with $a \leq^S_\bot b$ if and only if $a = b$ or $a = \bot$, for any $a, b \in S \cup \{\bot\}$. Let X and Y be two arbitrary sets.*

1. *Characterize the monotone functions $\mathcal{F} \in Mon(X_\bot, Y_\bot)$ with the property that $\mathcal{F}(\bot) \neq \bot$;*
2. *Are there any monotone functions $\mathcal{F} \in Mon(X_\bot, Y_\bot)$ such that $\mathcal{F}(x) = \bot$ for some $\bot \neq x \in X$?*
3. *For this item only, suppose that X and Y are finite:*
   (a) *How many partial functions are there in $X \rightharpoonup Y$?*
   (b) *How many (not necessarily monotone) total functions are there in $X_\bot \to Y_\bot$?*
   (c) *How many monotone functions are there in $Mon(X_\bot, Y_\bot)$?*
4. *Show that there is a bijective correspondence between partial functions in $X \rightharpoonup Y$ and monotone functions in $Mon(X_\bot, Y_\bot)$. Give an example of a total function in $X_\bot \to Y_\bot$ which does not correspond to a partial function in $X \rightharpoonup Y$.*

Monotone functions *preserve chains*, that is, $\{\mathcal{F}(d_n) \mid n \in \mathbb{N}\}$ is a chain in $(D', \sqsubseteq')$ whenever $\{d_n \mid n \in \mathbb{N}\}$ is a chain in $(D, \sqsubseteq)$. Moreover, if $(D, \sqsubseteq)$ and $(D', \sqsubseteq')$ are cpos then for any chain $\{d_n \mid n \in \mathbb{N}\}$ in $(D, \sqsubseteq)$, we have $\sqcup \mathcal{F}(d_n) \sqsubseteq' \mathcal{F}(\sqcup d_n)$. Indeed, since $\mathcal{F}$ is monotone and since $d_n \sqsubseteq \sqcup d_n$ for each $n \in \mathbb{N}$, it follows that $\mathcal{F}(d_n) \sqsubseteq' \mathcal{F}(\sqcup d_n)$ for each $n \in \mathbb{N}$. Therefore, $\mathcal{F}(\sqcup d_n)$ is an upper bound for the chain $\{\mathcal{F}(d_n) \mid n \in \mathbb{N}\}$. The rest follows because $\sqcup \mathcal{F}(d_n)$ is the lub of $\{\mathcal{F}(d_n) \mid n \in \mathbb{N}\}$.

Note that $\mathcal{F}(\sqcup d_n) \sqsubseteq' \sqcup \mathcal{F}(d_n)$ does not hold in general. Let, for example, $(D, \sqsubseteq)$ be $(\mathbb{N} \cup \{\infty\}, \leq)$, $(D', \sqsubseteq')$ be $(\{0, \infty\}, 0 \leq \infty)$, and $\mathcal{F}$ be the monotone function taking any natural number to 0 and ∞ to ∞. For the chain $\{n \mid n \in \mathbb{N}\}$, note that $\sqcup n = \infty$, so $\mathcal{F}(\sqcup n) = \infty$. On the other hand, the chain $\{\mathcal{F}(n) \mid n \in \mathbb{N}\}$ is stationary in 0, so $\sqcup \mathcal{F}(n) = 0$. Therefore, $\mathcal{F}(\sqcup n) = \infty \not\leq 0 = \sqcup \mathcal{F}(n)$.

Recall that one can think of the lub of a chain as the "limit" of that chain. The following definition is inspired from the analogous notion of continuous function in mathematical analysis, which is characterized by the property of preserving limits:

Definition 15. *A monotone function $\mathcal{F} : (D, \sqsubseteq) \to (D', \sqsubseteq')$ is **continuous** if and only if $\mathcal{F}(\sqcup d_n) \sqsubseteq' \sqcup \mathcal{F}(d_n)$, which is equivalent to $\sqcup \mathcal{F}(d_n) = \mathcal{F}(\sqcup d_n)$, for any chain $\{d_n \mid n \in \mathbb{N}\}$ in $(D, \sqsubseteq)$. Let $Cont((D, \sqsubseteq), (D', \sqsubseteq'))$ denote the set of continuous functions from $(D, \sqsubseteq)$ to $(D', \sqsubseteq')$. These notations are extended as expected when the posets are CPOs or BCPOs; e.g., $Cont((D, \sqsubseteq, \bot), (D', \sqsubseteq', \bot'))$.*

Exercise 23. *(See also Exercise 22). $Mon(X_\perp, Y_\perp) = Cont(X_\perp, Y_\perp)$, and this set of total functions bijectively corresponds to the set of partial functions $X \rightharpoonup Y$.*

Note that continuous functions in $Cont((D, \sqsubseteq, \perp), (D', \sqsubseteq', \perp'))$ need not take $\perp$ to $\perp'$ in general. In fact, as seen shortly in Section 2.9.4, the interesting continuous functions do not have that property.

Proposition 5. *$Cont((D, \sqsubseteq, \perp), (D', \sqsubseteq', \perp'))$ can be naturally endowed with a BCPO structure, where*

- *Its partial order is defined as $f \preceq g$ if and only if $f(d) \leq g(d)$ for all $d \in D$;*
- *Its bottom element is defined as the function $\perp$ with $\perp(d) = \perp'$ for all $d \in D$.*

Exercise 24. *$(Cont(X_\perp, Y_\perp), \preceq, \perp)$ and $(X \rightharpoonup Y, \preceq, \perp)$ are isomorphic BCPOs.*

2.9.4 The Fixed-Point Theorem

Any monotone function $\mathcal{F} : (D, \sqsubseteq, \perp) \to (D, \sqsubseteq, \perp)$ defined on a BCPO to itself admits an implicit and important chain, namely $\perp \sqsubseteq \mathcal{F}(\perp) \sqsubseteq \mathcal{F}^2(\perp) \sqsubseteq \cdots \sqsubseteq \mathcal{F}^n(\perp) \sqsubseteq \cdots$, where $\mathcal{F}^n$ denotes n compositions of $\mathcal{F}$ with itself. The next theorem is a key result, which has major implications in many areas of mathematics and computer science, in particular in denotational semantics (Section 3.7):

Theorem 1. ***(The fixed-point theorem)*** *Let $(D, \sqsubseteq, \perp)$ be a BCPO, let $\mathcal{F} : (D, \sqsubseteq, \perp) \to (D, \sqsubseteq, \perp)$ be a continuous function, and let $fix(\mathcal{F})$ be the lub of the chain $\{\mathcal{F}^n(\perp) \mid n \in \mathbb{N}\}$. Then $fix(\mathcal{F})$ is the least fix-point of $\mathcal{F}$.*

Proof. We first show that $fix(\mathcal{F})$ is a fix-point of $\mathcal{F}$:

$$\begin{array}{rcl} \mathcal{F}(fix(\mathcal{F})) & = & \mathcal{F}(\bigsqcup_{n\in\mathbb{N}} \mathcal{F}^n(\perp)) \\ & = & \bigsqcup_{n\in\mathbb{N}} \mathcal{F}^{n+1}(\perp) \\ & = & \bigsqcup_{n\in\mathbb{N}} \mathcal{F}^n(\perp) \\ & = & fix(\mathcal{F}). \end{array}$$

Next we show that $fix(\mathcal{F})$ is the least fix-point of $\mathcal{F}$. Let d be another fix-point of $\mathcal{F}$, that is, $\mathcal{F}(d) = d$. We can show by induction that $\mathcal{F}^n(\perp) \sqsubseteq d$ for any $n \in \mathbb{N}$: first note that $\mathcal{F}^0(\perp) = \perp \sqsubseteq d$; assume $\mathcal{F}^n(\perp) \sqsubseteq d$ for some $n \in \mathbb{N}$; since $\mathcal{F}$ is monotone, it follows that $\mathcal{F}(\mathcal{F}^n(\perp)) \sqsubseteq \mathcal{F}(d) = d$, that is, $\mathcal{F}^{n+1}(\perp) \sqsubseteq d$. Thus d is an upper bound of the chain $\{\mathcal{F}^n(\perp) \mid n \in \mathbb{N}\}$, so $fix(\mathcal{F}) \sqsubseteq d$. □

Let us next discuss some interesting applications of the fixed-point theorem.

A first application we discuss here is to show that recursively defined (mathematical) functions are correct. Consider the following common definition of the factorial:

$$f(n) = \begin{cases} 1 & \text{, if } n = 0 \\ n * f(n-1) & \text{, if } n > 0 \end{cases}$$

How does one know that such a mathematical object, i.e., a function satisfying the above property, actually exists and is unique, as tacitly assumed? To see that such a concern is justified, replace "$n * f(n-1)$ when $n > 0$" by "$n * f(n-2)$ when $n > 1$" in the above and note that there are infinitely many functions satisfying the above property, or replace "$n * f(n-1)$" by "$n * f(n+1)$" and note

that there is no function with the property above. According to the fixed-point theorem, since the function $\mathcal{F}$ defined on the set of partial functions $\mathbb{N} \rightharpoonup \mathbb{N}$ to itself as

$$\mathcal{F}(g)(n) = \begin{cases} 1 & \text{, if } n = 0 \\ n * g(n-1) & \text{, if } n > 0 \text{ and } g(n-1) \text{ defined} \\ \text{undefined} & \text{, if } n > 0 \text{ and } g(n-1) \text{ undefined} \end{cases}$$

is continuous, it has a least fixed point. We thus can take $f = fix(\mathcal{F})$, and get

$$f(n) = \mathcal{F}(f)(n) = \begin{cases} 1 & \text{, if } n = 0 \\ n * f(n-1) & \text{, if } n > 0 \text{ and } f(n-1) \text{ defined} \\ \text{undefined} & \text{, if } n > 0 \text{ and } f(n-1) \text{ undefined} \end{cases}$$

One can easily show (by induction) that f is defined everywhere, so the above is equivalent to the original definition of f. Since f is total, it is the *unique* fixed point of $\mathcal{F}$: any fixed-point f' of $\mathcal{F}$ obeys $f \preceq f'$, so f' is also defined everywhere and equal to f.

Exercise 25. *Prove that the function $\mathcal{F}$ defined above satisfies the hypothesis of Theorem 1, in particular show that $\mathcal{F}$ is continuous. What if one replaces "$n * g(n-1)$ when $n > 0$" by "$n * g(n-2)$ when $n > 1$" or "$n * g(n-1)$" by "$n * g(n+1)$"?*

A second application of the fixed-point theorem that we discuss here is to show why and how recursively defined languages admit unique solution. Any context-free language over a possibly infinite alphabet, or set of *terminals*, can be defined as the least fixed point of some continuous operator on the power set of the set of words over the given alphabet. Let for instance the alphabet be $A = Var \cup \mathbb{Z} \cup \{+, -, *\}$, where $\mathbb{Z}$ is the set of integers and Var is a set of variables, and consider the following context-free grammar giving the syntax for arithmetic expressions:

$$Exp \quad ::= \quad Z \quad | \quad Var \quad | \quad Exp + Exp \quad | \quad -Exp \quad | \quad Exp * Exp$$

There are many different ways to describe or construct the language of arithmetic expressions corresponding to the grammar above, which we do not discuss here. What we want to highlight here is that it can also be described as the least fixed point of the following continuous function, where A^* is the (infinite) set of finite words with letters in A:

$$\mathcal{F} : (\mathcal{P}(A^*), \subseteq, \varnothing) \to (\mathcal{P}(A^*), \subseteq, \varnothing), \text{ where}$$

$$\mathcal{F}(L) \;=\; Var \;\cup\; \mathbb{Z} \;\cup\; L\{+\}L \;\cup\; \{-\}L \;\cup\; L\{*\}L.$$

We used the notation $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. Notice that the iterations $\mathcal{F}(\varnothing)$, $\mathcal{F}^2(\varnothing)$, ... correspond to the one-step, two-steps, ... derivations applying the grammar's productions.

Exercise 26. *Prove that the function $\mathcal{F}$ above satisfies the hypothesis of Theorem 1, in particular that $\mathcal{F}$ is continuous. Give a general means to associate such a function to any context-free grammar, so that its least fixed point is precisely the language of the grammar.*

We shall later see that the fixed-point theorem above can also be used to give denotational semantics both to `while` loops (in Section 3.7) and to the recursive construct μ (in Section 4.7.3). Fixed-points in general and the theorem above in particular have many applications in both computer science and mathematics. At the time this book was being written, there was even a conference dedicated to fixed points, the *International Conference on Fixed Point Theory and Its Applications*.

sorts:

Var_{CPO}, CPO

subsorts:

Var_{CPO}, $Bool$ $<$ CPO

operations:

$$\begin{array}{rcl} \bot & : & \{*\} \to [CPO] \quad // \text{ “undefined” constant of kind } [CPO] \\ \texttt{fun}_{CPO\_}\texttt{->}\_ & : & Var_{CPO} \times [CPO] \to [CPO] \\ \texttt{app}_{CPO} & : & [CPO] \times [CPO] \to [CPO] \\ \texttt{fix}_{CPO} & : & [CPO] \to [CPO] \\ \texttt{if}_{CPO} & : & [CPO] \times [CPO] \times [CPO] \to [CPO] \end{array}$$

equations:

$$\begin{array}{rcl} \texttt{app}_{CPO}(\texttt{fun}_{CPO} V \texttt{->} C,\ C') & = & C[C'/V] \\ \texttt{fix}_{CPO}(\texttt{fun}_{CPO} V \texttt{->} C) & = & C[(\texttt{fun}_{CPO} V \texttt{->} C)/V] \\ \texttt{if}_{CPO}(\texttt{true}, C, C') & = & C \\ \texttt{if}_{CPO}(\texttt{false}, C, C') & = & C' \end{array}$$

Figure 2.4: Equational framework for cpos and fixed-points (assuming Booleans and substitution).

2.9.5 Equational Framework for Fixed-Points

As seen above, fixed-points can be complex mathematical objects, such as functions or infinite sets of words, which may require complex means to define, understand or visualize. In many cases the only option is to “experiment” with the fixed-point, such as to evaluate it on a given input when it is a function, or to check whether it contains a certain element when it is a set. If one wants to formally reason about or to compute fixed points, then one needs to formally define the body of mathematics involved in the definition of the function for which the fixed point is desired. For example, the factorial above requires some formal definition of the domain of integers, at least with multiplications and subtraction, as well as some formal definition of the domain $Int \rightharpoonup Int$ of partial functions from integers to integers (in order to define $\mathcal{F}$). Similarly, the CFG language above requires the formal definition of sets with union and possibly other operations on them, and of the domain of finite words over a given alphabet, with at least a concatenation operation on them, as well as a combination of the two, where concatenation is extended to sets of words. Even though in theory we assume the entire arsenal of mathematics, in practice we only have a limited fragment of mathematics available for formal definitions of fixed points. In this section we define such a limited framework, but one which is quite common and particularly useful for the application of fixed-points in denotational semantics (Section 3.7).

Figure 2.4 shows a simple equational theory of a minimal framework for formally defining fixed-points. We call it minimal because it only includes support for defining and applying functions, together with a simple extension for defining cases. One will need to add further extensions to this framework in order to support definitions of more complex fixed-points. For defining cases, we assumed that Booleans are already defined. For convenience, from here on we also assume that integers are already defined, together with all the desired operations on both Booleans and integers. To make it clear from which mathematical domain each operation or relational symbol comes, we add the domain as a subscript to the operator or relational symbol. For example, $+_{Int}$ is the addition on integers, $\leq_{Int}$ is the comparison of integers which evaluates to a Boolean value, and $==_{Bool}$ is a

generic equality test over terms of any sort (if one's framework does not allow for polymorphic operations then one may need to define such an equality test operation for each sort). The functions, their application and their fixed-points have been defined like in untyped call-by-value λ-calculus (see Section 4.4) extended with the recursion μ construct (see Section 4.7), using an appropriate notion of substitution (see Section 4.4.3). How these are defined is not important here, so we refer the interested reader to the above-mentioned sections for details. What is important here is that the equational theory in Figure 2.4 gives us a starting point for formally defining fixed-points.

For simplicity, we considered only one major syntactic category in the equational theory in Figure 2.4, namely *CPO*, for complete partial orders. Basic CPOs other than that of Booleans $Bool_\perp$, e.g., the CPO of integers $Int_\perp$, need to be subsorted to *CPO* if one wants to include them. We concentrate on functional domains here, that is, on CPOs whose elements are partial functions, with $\perp$ the partial function which is undefined everywhere. We sticked to our naming convention and thus we tagged all the newly introduced operation with *CPO*, to distinguish them from possibly homonymous operations from included domains. We used the membership equational logic notation for *kinds*, as sorts in square brackets, to denote the fact that operations may by undefined. The most obvious use of it is for the result sort of the $\perp$ constant. Domains can also chose to generate undefined terms, such as attempts to perform a division by zero in the domain of integers. The four equations in Figure 2.4 are straightforward, capturing the meaning of the four language constructs in terms of substitution; here we assume substitution given, as explained above.

We can now use the constructs in Figure 2.4 to define fixed-points. For example, the factorial function discussed above can be defined as the term below, say `factorial`:

$$\texttt{fix}_{CPO}(\texttt{fun}_{CPO}\ g\ \texttt{->}\ \texttt{fun}_{CPO}\ n\ \texttt{->}\ \texttt{if}_{CPO}(n \texttt{==}_{Bool} 0, 1, \texttt{if}_{CPO}(n \texttt{>}_{Int} 0, n \texttt{*}_{Int} \texttt{app}_{CPO}(g, n \texttt{-}_{Int} 1), \perp)))$$

Using the provided equations, one can prove, for example, that `factorial`$(3) = 6$.

Exercise 27. *Define the famous Ackermann function*

$$a(m,n) = \begin{cases} n+1 & \textit{if } m = 0 \\ a(m-1,1) & \textit{if } m > 0 \textit{ and } n = 0 \\ a(m-1,a(m,n-1)) & \textit{if } m > 0 \textit{ and } n > 0 \end{cases}$$

as a fixed-point, then represent it in the formal language above in two different ways: first as a curried function taking one argument and producing a function taking another argument and then producing an integer results, and second as a function taking a pair argument. For the latter, extend the equational theory in Figure 2.4 with a pair construct and corresponding projection operations.